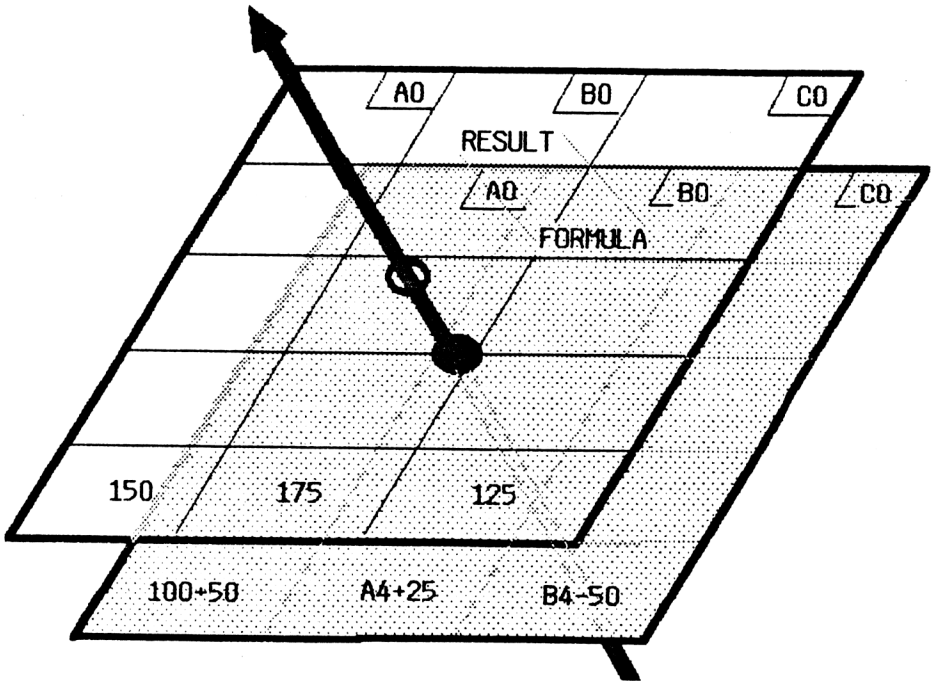
POCKET COMPUTER NEWSLETTER

© Copyright 1983 POCKET COMPUTER NEWSLETTER



HOW IT OPERATES

SUPER SST can be conceptually viewed as two planes of information as shown in the adjacent drawing. One plane (the lower one in the diagram) holds *data*. This data may take the form of actual numerical values or it may be expressed as formulas. When expressed as formulas the "variables" are the contents of other cells. More precisely, the variables are the *values* of cells that are in the second plane. This plane is referred to as the *results* matrix.

When you select the DEF (define) option from the program's main memory, you place the program in a mode that enables you to enter information onto the data (lower) plane. When a cell is activated, any previous contents of the cell in the data matrix are displayed. The cell contents may be initialized or altered by entering new data. To speed the entry process, the program will automatically step down a column or across a row each time the ENTER key is pressed. Whenever a change is made to a cell in the data matrix, the corresponding position in the results matrix is immediately changed to the "minus nines" state. This serves as a reminder that the cell has not been recalculated. Additionally, a "recalculate" flag is raised to indicate that the results matrix should be processed for possible changes.

When a formula is entered into the data matrix, a machine language routine is used to verify that the information can be properly "parsed." If not, then the operator is advised to re-enter the information. *SUPER SST* uses a true algebraic parsing system so that mathematical operators may be processed in a heirarchy similar to that used in BASIC. This makes it easier for users to work with the program. Checking for proper syntax during the inputting process relieves the parser of that process during the recalculation procedure. This greatly speeds the recalculation process.

When the DIS (display) option is selected from the main menu, the program processes the numbers and formulas in the data matrix and "projects" the answers into the results (top plane) matrix. The bulk of this processing is accomplished using machine language routines which provide the ultimate in processing speed. Once a recalculation has taken place, the operator is given the ability to scroll about the results matrix and view the outputs.

Data Matrix

* STOCK RECORDER	COL A: DATE	COL B: PRICE	COL C: QTY	COL D: STK COST	COL E: CMSNS	COL F: NET COST	COL G:
ROW 0: TX NYSE	8304.20	34.625	500	B0*C0	215	D0+E0	8308.05
ROW 1: XON NYSE	8304.22	33.25	100	B1*C1	86.25	D1+E1	8306.02
ROW 2: CMB NYSE	8304.28	55.625	300	B2*C2	188.75	D2+E2	8305.12
ROW 3: AAPL OTC	8306.06	62.625	200	B3*C3	155	D3+E3	8308.26

Results Matrix

* STOCK RECORDER	COL A: DATE	COL B: PRICE	COL C: QTY	COL D: STK COST	COL E: CMSNS	COL F: NET COST	COL G:
ROW 0: TX NYSE	8,304.20	34.63	500.00	17,312.50	215.00	17,527.50	
ROW 1: XON NYSE	8,304.22	33.25	100.00	3,325.00	86.25	3,411.25	
ROW 2: CMB NYSE	8,304.28	55.63	300.00	16,687.50	188.75	16,876.25	
ROW 3: AAPL OTC	8,306.06	62.63	200.00	12,525.00	155.00	12,680.00	

STOCK PORTFOLIO TEMPLATE

Your *SUPER SST* program can readily be adapted to serve as a stock portfolio analyzer. If your holdings are modest (10 or less issues at a time) then the status of your stocks can be updated and summarized in just a few moments a day.

The accompanying diagram provides a template that can be used to maintain a computerized stock portfolio. This discussion will highlight key aspects of this application.

First, note that column labels have been assigned to all 14 column headings. This starts at column A (DATE) and extends through column N (PERCENT). These headings are essentially self-explanatory.

You may title your spreadsheet as you see fit. (STOCK RECORDER was used as a title for the example template.)

Place the names of the stocks you plan to keep track of as row labels. In the example, the ticker tape symbol and the appropriate trading exchange were used.

The use of each column is described as follows:

Column A: Date. In order to fool the program into displaying this in an understandable form, enter it as YYMM.DD (two digit year, month, then a decimal point and the day-of-the-month as two digits). Alternately, drop the year entirely, or use just the last digit of the year, thus: YMM.DD.

Column B: Price. This is the price at which you purchased the stock on a per-share basis. Since stock prices are usually quoted to the nearest 1/8th of a dollar, enter the fractional portion as an equivalent decimal value. For instance 34-5/8 as 34.625. *Note that SUPER SST will accept the fractional portion to 3 or 4 decimal places when it is inputted and store this internally. However, when outputted, this value will appear as rounded to 2 decimal places!*

Column C: Quantity. Enter the number of actual shares purchased for each stock you are tracking.

Column D: Stock Cost. This is a formula that the spreadsheet will automatically calculate based on the contents of columns B and C. Use the form given in the template.

Column E: Commissions. Just enter the actual commissions paid at the time of stock purchase for the quantity of stock specified in column C.

Column F: Net Cost. Another simple formula, based on the contents of columns D and E is used to calculate the total purchase cost of the stock.

Column G: Date Sold. Use the same YYMM.DD or similar format as that used in column A. Naturally, this and subsequent columns are not utilized until you sell a stock *or you desire to analyze the projected results if you were to sell!*

DTE SOLD	COL H: SALE PRC	COL I: STK SALE	COL J: CMSNS	COL K: NET SALE	COL L: TTL CMSN	COL M: NET P/L	COL N: PERCENT
	35.50	H0*C0	235.5	I0-J0	E0+J0	K0-F0	M0/F0*100
	33.50	H1*C1	101.50	I1-J1	E1+J1	K1-F1	M1/F1*100
	61.50	H2*C2	230.50	I2-J2	E2+J2	K2-F2	M2/F2*100
	30.75	H3*C3	182.50	I3-J3	E3+J3	K3-F3	M3/F3*100

DTE SOLD	COL H: SALE PRC	COL I: STK SALE	COL J: CMSNS	COL K: NET SALE	COL L: TTL CMSN	COL M: NET P/L	COL N: PERCENT
8,308.05	35.50	17,750.00	235.50	17,514.50	450.50	-13.00	-0.07
8,306.02	33.50	3,350.00	101.50	3,248.50	187.75	-162.75	-4.77
8,305.12	61.50	18,450.00	230.50	18,219.50	419.25	1,343.25	7.96
8,308.26	30.75	6,150.00	182.50	5,967.50	337.50	-6,712.50	-52.94

Column H: Sale Proceeds. This is the per-share sale price. The same comments regarding fractional inputs and display rounding as mentioned for column B also apply here.

Column I: The formula used here will multiply the sale price times the number of shares taken from column C. Note that it is assumed that you sell as many shares as you originally purchased. (If this is not the case, make multiple row entries, one for each block of stock traded.)

Column J: Sale Commissions. Enter the total commissions paid for the selling transaction.

Column K: Net Sale. The formula in this column will yield the sale price less sales commissions.

Column L: Total Commissions. The formula here combines buying and selling commissions. You might be surprised how hefty the total buy/sell broker's take can be. It is worth keeping an eye on. Are you trading enough to take advantage of a discount broker?

Column M: Net Profit/Loss. This is the key formula that tells you whether the whole effort was worthwhile!

Column N: Percentage. This formula gives you the percent gain or loss on the total transaction after commissions. Sometimes this figure is easier to digest and is more meaningful than the raw dollar totals. (The whole object of the investment game is to keep the calculated figures in this column on the positive side!)

Variations

If you have a small portfolio and only trade a few times a year, then you can pretty much use the template as presented. Naturally, you save the status of the portfolio on tape for updating as desired.

If you manage a fairly active portfolio and like to follow stocks on a daily basis, then take note of the following: You can plug current prices into column H whenever desired. Change column J to a rough approximation of sale commissions using an appropriate formula, such as 3% of column I. Then use the results of columns I, M and N as a daily barometer.

Finally, you might want to use the 10th row for summing columns instead of tracking a tenth stock. For instance, placing the formula F0=Fn (that is, summing all the active rows in column F) at cell position F9 would give you a total cost for all the stocks in your portfolio. Summing other columns would provide similar summaries.

The accompanying diagram of the Results Matrix illustrates the type of output one will receive when using the illustrated template.

Here's hoping all your percentage figures are on the up side!

PERFORMING SPECIAL OPERATIONS

Have you ever wished you could use transcendental functions such as the tangent, exponentiation and logarithmic operations, as part of a spreadsheet calculation? Take heed: if your application has vital need of such capabilities, if the format you use can be essentially fixed, and if you are willing to do some programming work yourself, then you may be able to "integrate" such functions into the operation of *SUPER SST!*

If you have read the article on page 1 of this issue, then you know that *SUPER SST* projects the results of calculations into the *results* matrix. The cells of this matrix are simply elements in a two-dimensional array named R. If you list line 20 of the BASIC portion of the program, you can see that this array is dimensioned as R(13,9). Thus, it contains 14 times 10 elements (since there is a zero element in each dimension) corresponding to the 14 columns and 10 rows that make up a *SUPER SST* spreadsheet.

Guess what? You can access any cell or element in that array. You could, for instance, run a spreadsheet calculation, then go in and manipulate any value in that array.

In fact, if you only need to, for instance, multiply some value by a transcendental function, then that is the easiest approach to take. Calculate the spreadsheet. Stop the program (by pressing the BREAK key when it is in the menu mode). Access the desired array element by specifying the appropriate subscripts. (Columns A through N are represented by the first subscript 0 through 13. Rows 0 through 9 correspond directly to the second subscript.) Multiply the array element by the desired transcendental function, such as LOG 2. Assuming this is done in the direct execute mode, the answer for the calculation will appear on the LCD when the ENTER key is tapped.

This particular approach has the added benefit of leaving the original value unchanged. The obvious drawback is that the answer does not become part of the spreadsheet. Unless, of course, you invoke a LET statement so that R(i,j)=R(i,j)*[function], etc. In this latter case, however, any change to the cell would not be reflected by other cells that might reference that location within the spreadsheet.

In order to have such a special operation affect other cells in the matrix, the function must be performed at the time that particular cell is evaluated by *SUPER SST*. It is possible to do this, but a great deal of speed in the overall operation of the program will be sacrificed. Never-the-less, this may be a worthwhile trade-off for some users.

Line 4050 in the BASIC coding of *SUPER SST* sets up a nested loop that scans the data matrix. The loop variables I and J access the columns and rows respectively. Thus, when I=0 and J=0, cell position A0 is processed. I=0, J=1 causes position A1 to be analyzed. I=13, J=9 is how the pointers would appear as cell position N9 was processed.

Line 4060 calls the machine language routines that perform the actual calculations for each cell.

Line 4080 advances the row and column loop variables. *SUPER SST* goes down rows within a column before advancing to analyze the next column. Thus, J advances from 0 through 9 while I is held at 0, then J is reset, I is advanced to 1, and the process repeats as J again advances to 9, etc.

How convenient! Inserting a line numbered 4070 is an ideal way to add supplemental capability! Suppose, for instance, that you wanted to multiply every element in column A by the function LOG 2. The following would accomplish the objective:

```
4070 IF I=0 LET R(I,J)=R(I,J)*LOG 2
```

Note that the trick here is to test the loop variables in order to ascertain which cells are being processed. You then perform your desired transcendental function operations only on the desired cell(s). Suppose, for instance, that there was only one cell in the entire matrix that required your special treatment: C5. Then the test would appear as:

```
4070 IF I=2 AND J=5 LET R(I,J)=R(I,J)*LOG 2
```

Of course, several or a series of such tests can be inserted in the line range 4061 - 4079 if desired. The more insertions, the more speed degredation.

Cautions: This technique should only be used to invoke transcendental functions that operate with numeric values. Do ***not*** *attempt to assign values to variables other than elements of the R array. The machine language routines used in the program utilize various variable locations. Should you alter one of these, the program will quite likely lock up, resulting in the loss of all data stored in memory.*

Information about this technique is provided solely for educational and experimental purposes. Application of this information is solely at your own risk. The implementation of these techniques are not considered to be within the normal intended use of this program and are not supported in any fashion by the program's publisher. Should you care to experiment with these procedures, you are ***strictly on your own!***

P.O. Box 232, Seymour, CT 06483